(2)

219800-10-T

Programmer's Manual

MAX VIDEO NEIGHBORHOOD PROCESSOR PIPELINE (MVNPP)

R.J. HORNER

JULY 1990

Prepared for:

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-5800

Contract No. 42-3638

ERIM
P.O. Box 8618
Ann Arbor, MI 48107-8618

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
219800-10-T		

4. Title and Subtitle	5. Report Date
MAX VIDEO NEIGHBORHOOD PROCESSOR PIPELINE Programmer's Manual	July 1990
	6. Performing Organization Code: ERIM
7. Author(s): Robert J. Horner	8. Performing Organization Report No.: 219800-10-T
9. Performing Organization Name and Address: Environmental Research Institute of Michigan, P.O. Box 8618, Ann Arbor, MI 48107-8618	10. Work Unit No.
	11. Contract or Grant No.: 42-3638
12. Sponsoring Agency Name and Address: Scandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-5800	13. Type of Report and Period Covered: Programmer's Manual, Sept. 1989 - Dec. 1990
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This manual describes the functional capabilities and programming of ERIM's Max Video* Neighborhood Processor Pipeline (MVNPP) board. It is intended for the user who wishes to understand the MVNPP's purpose and functionality, and the system programmer who must have detailed information on control registers and operation in order to operate the MVNPP effectively.

The manual is organized into four chapters and an appendix. In Chapter I, a high-level description of the MVNPP functions is given. Chapter II is devoted to describing the status and control registers that constitute the software interface to the MVNPP. In Chapter III, programming examples of common system functions are presented. Chapter IV details how to configure the MVNPP for inclusion in different image processing environments. Appendix A details differences between multiple versions of the MVNPP, which affect the operation or programming.

This manual describes only MVNPP board operation and programming. For details of programming the Cyto-HSS stages, see the Cyto-HSS Stage Programmer's Manual (ERIM document number IPTL-89-294). Programming of a complete system including the MVNPP board will be broadly discussed, but for detailed information concerning programming of other boards please see the appropriate board programmer's manual.

* Max Video is a trademark of Datacube, Incorporated

17. Key Words	18. Distribution Statement
Computer Programming Programming Manual	Approved for public release; distribution is unlimited.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	29	

CONTENTS

FIGURES

1.0 Functional Description

The CytocomputerMVNPP is a low-cost hardware module that allows cellular logic image processing to be added to almost any VMEbus system. In particular, the MVNPP is compatible with the SUN^{tm} family workstations, Datacube Incorporated's Max Video bus family of boards, and ERIM's Cyto-HSS image processing workstations. The Cytocomputer MVNPP includes the following capabilities and functions:

- Parallel processing of eight-bit images at over 16,000 MIPS equivalent performance.
- Ability to process images up to 2048 pixels wide and any length.
- A pipeline of up to 16 Cyto stage processors.
- Maxbus input and output ports for the pipeline.
- Full Region Of Interest (ROI) timing support.
- High-speed Maxbus-based programming of the processors.
- A Cyto-HSS image bus interface to the pipeline.
- A Cyto-HSS stage backplane interface to the pipeline.
- Direct VME bus access to the pipeline.
- A diagnostic port with full VMEbus access to stage control signals.
- Support for high-speed image transfers over the VMEbus.
- Simple cascading of multiple MVNPP modules.

The MVNPP may be used with ERIM's C4PL interactive image processing language to accelerate image processing over a software only implementation. It may be plugged into a Cyto-HSS workstation to provide long pipelines for operation with C4PL. The MVNPP can also be programmed and operated directly from "C" or other conventional languages in a Max Video environment.

Figure 1 is a block diagram of the MVNPP. The board combines multiple interfaces with a 16-stage pipeline to provide the high-speed processing advantages of Cyto technology to the most common VMEbus image processing configurations. For example, the Maxbus interfaces integrate the MVNPP into a widely used high-performance image processing system. Similarly, the Cyto-HSS interfaces allow the MVNPP to be used in ERIM's Cyto-HSS workstations. Finally, the VME data interface allows any computer with a VMEbus interface to efficiently program the stages and pass images through the pipeline. The following sections describe the MVNPP functional blocks in detail.

1.1 VMEbus Interface

The VMEbus interface provides direct access to the stage pipeline, as well as control and status registers that can select and operate the high-speed interfaces to the pipeline.

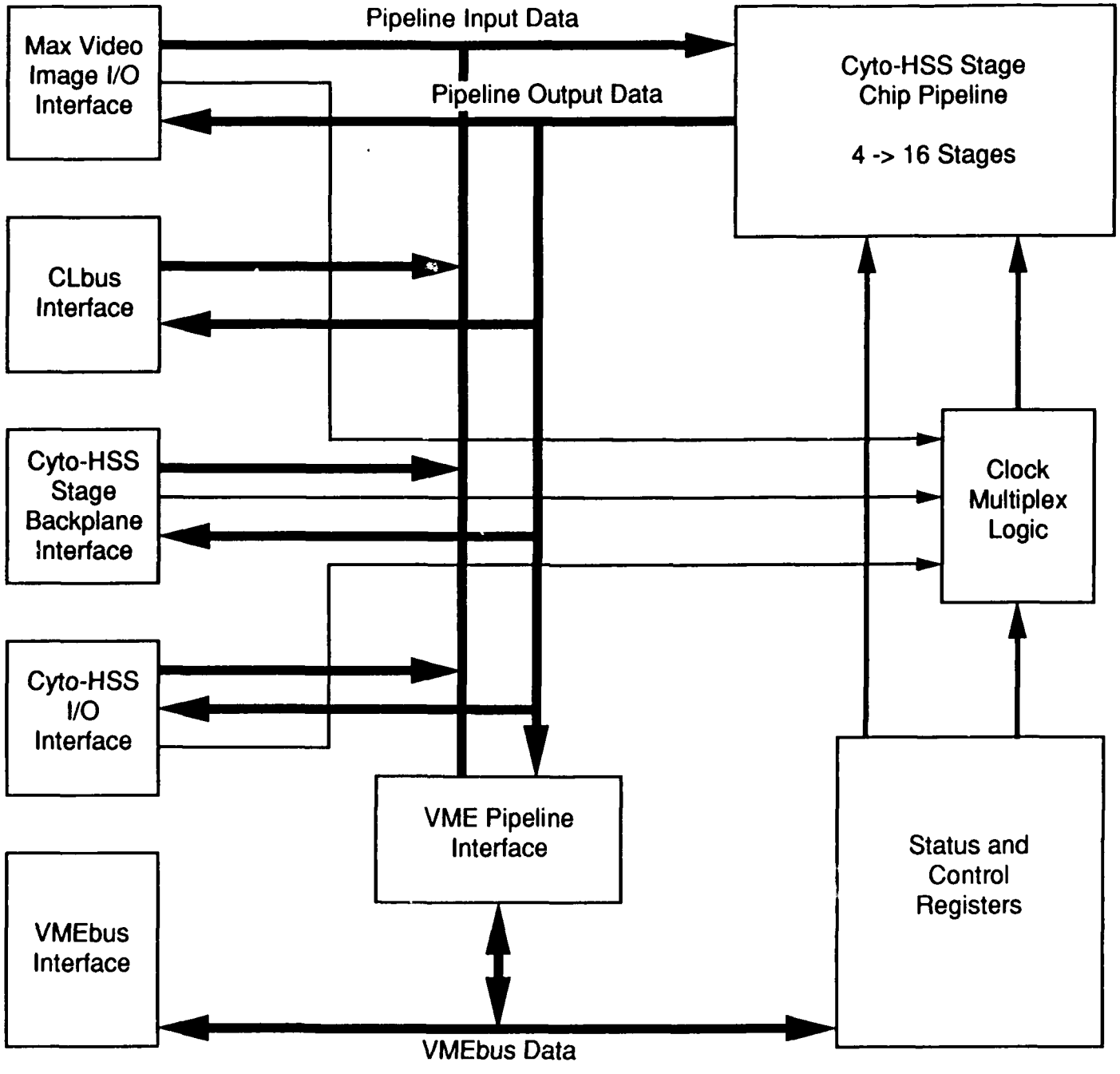

Figure 1. MVNPP Block Diagram

The VME interface conforms to the Rev.C bus specification and contains an A32/A24/A16 and D32/D16/D08(EO) slave responder. Only the pipeline image data port, however (see section 2.5), can be usefully accessed in any data width other than D16. Any byte or either word of the image data port may be accessed, as well as the entire long word. Accesses to any register except the pipeline image data port must be by word only to ensure correct operation.

The VME image data interface to the pipeline is designed to minimize the image circulation time through the pipeline. Long-word access is supported to provide the maximum bandwidth over the VMEbus, but byte and word accesses may also be used for images that are not long-word aligned. Also, a counter automatically generates the stage control signals required during an image circulation so that the processor need only write and read the image data. Finally, a first-in-first-out (FIFO) buffer is placed at the end of the pipeline.

The FIFO is 4096 pixels deep and can, therefore, hold at least two lines of image data even at the maximum supported line length. This allows a CPU to utilize built-in fast-move instructions to write an entire line of the image to the pipeline and then read an entire line from the pipeline, rather than having to write a byte and then read a byte. A two line FIFO is necessary to ensure that whole lines may be written to the FIFO even if the latency through the pipeline is not a multiple of the image line length.

The VMEbus interface to the pipeline takes precedence over all other (high-speed) interfaces to the pipeline. If another interface is enabled and a VME access of the pipeline is attempted, the other interface is temporarily disabled and the VME access proceeds. This could easily corrupt an image transfer that may be occurring on the other interface when the VME access begins. For this reason, the user must take care when accessing the pipeline from the VMEbus. In general, no VME accesses should occur while another interface is transferring an image through the stage pipeline. This should not cause a problem because a single program would generally control all the high-speed processing boards in the system and the programmer would obtain status from, for instance, the high-speed memory board if a processing cycle has been initiated.

1.2 Maxbus I/O Interface

The Maxbus image I/O interface allows images to be passed through the Cyto stage pipeline on the MVNPP at a rate of 10 Mpixel/second. The interface provides all the logic needed to support Maxbus Region Of Interest (ROI) image transfers on the input and output data paths. Any of the four Maxbus ROI timing sources may be selected to control the interface. Four 12-bit counters define the image position relative to the Maxbus HSync~ and VSync~ timing signals (see the Datacube ROIStore manual for details of ROI timing).

A FIFO is also connected to the end of the stage pipeline during Maxbus image transfers. This FIFO provides pipeline latency compensation needed to connect the stage pipeline into a Maxbus system. The latency effects of the FIFO will be discussed in Section 1.9 on the MVNPP latency.

1.3 Maxbus Programming Interface

This logic connects a high-speed coefficient loading bus (CLbus) to the stage pipeline to allow high-speed programming of the stages. The CLbus is based on the Max Video ROI data transfer. Two 8-bit Maxbus data paths are used to supply the 11 bits (8 data and 3 control) needed to program the stage pipeline at one 11-bit word per ROI clock. The stage programs maybe stored in a ROIstore as 16-bit words and loaded into the pipeline with a single ROI image transfer cycle. The MVNPP CLbus interface allows selection of any one of the four ROI timing buses to control the stage program data loading. Four counters in the CLbus interface (similar to the Maxbus I/O interface counters) select the valid programming data on the ROI data bus during a program transfer.

1.4 Cyto-HSS I/O Interface

This interface circuitry allows the MVNPP to be directly connected into a Cyto-HSS system card cage in one of the VMEbus slots. Pipeline input comes from the Port board via a front panel cable. Likewise, the pipeline output goes to the Combiner board via a front panel cable. However, unlike previous generations' processing stages, the MVNPPs does not need to plug into a separate stage card cage; they can reside in any open VMEbus slot.

There is one minor hardware incompatiblilty when using the MVNPP connected directly to the Port and Combiner (i.e., not in a stage cardcage). One of the low-level test capabilities of the Combiner allows back-driving the pipeline output cable to test the Combiners pipeline receiver logic. Test software that utilizes this feature must be modified to explicitly disable the MVNPPs output drivers before running the test. This may be done by selecting the VMEbus interface to drive the pipeline (see Section 2.2.4). The existing Cyto-HSS stages and Combiner performed this function implicitly but the method used is incompatible with the cascading logic needed to connect multiple MVNPP boards together in a VMEbus card cage.

The Cyto-HSS I/O circuitry also allows multiple MVNPP boards to be physically cascaded and logically treated as a single board with a longer pipeline. This cascading is effective whether the pipeline is driven from the VMEbus, the Cyto-HSS interface, or the Max Video interfaces. Pipeline data and control lines daisy-chain from one board to the next (internal to the cascade set) via Cyto-HSS connectors. Section 1.8 provides a more detailed discussion of the MVNPP board cascading features and restrictions.

1.5 Cyto-HSS Stage Backplane Interface

This interface allows the MVNPP to be operated from a Cyto-HSS stage backplane. The MVNPP is indistinguishable in software from 16 chip stages on separate boards. If the chip stage command extensions are not used, the MVNPP is indistinguishable from 16 gate array based stage boards. This allows the existing Cyto-HSS workstations to be used for high-speed testing of the stages on the MVNPP via the HSNPSTP test software. This also allows longer pipelines to be implemented in the Cyto-HSS workstations.

1.6 Neighborhood Processor Stage Pipeline

This section implements up to a 16 stage pipeline with the associated stage line-delay memories. Twelve of the 16 stages are installed on a daughter card attached to the main MVNPP board. The daughter card may be removed and the connection bypassed to allow operation with fewer stages. Likewise, either the mother or daughter card maybe partially populated with the jumpers bypassing the absent stages.

1.7 Status and Control Registers

The control and status registers provide overall control of the pipeline board and monitoring of error and other status information. These registers are described in detail in Chapter II.

1.8 Multiple Board Cascading

The cascading logic of the MVNPP has a simple goal: to make the system programmer's job easier. The circuitry works to ensure that there is no way (in software) to distinguish an N-board cascaded set from a single board with $N * 16$ stages. To achieve this, several board configuration requirements must be met:

1. All boards must be configured at the same VMEbus address.
2. All boards of the set must be daisy-chain cabled together using the Cyto-HSS I/O connectors (see Section 4.2).
3. For Max Video I/O, all boards must be connected to the same ROI timing cable.
4. For either Max Video or Cyto-HSS I/O, the input and output cables must be connected to the first and last boards of the set respectively,

Detailed descriptions of the MVNPP connectors can be found in Chapter IV. The above rules follow naturally from the idea that a cascaded board set is equivalent to a single board with a longer pipeline. The daisy-chain cables and common VMEbus address form a cascaded board set. The other rules ensure that the cascaded pipeline input and output are correctly connected.

The cascading logic functions by altering the selection of the data I/O to/from the pipeline on a given board, depending on whether the board is at the beginning, middle, or end of the cascaded pipeline. The logic also controls accesses to the VMEbus accessible registers to ensure that register writes occur on all boards, but reads only return data from the last board. Additional logic ensures that the last board contains all the valid status information. Therefore, software will only see one set of MVNPP board registers, and identical programming for a single board and a cascaded board set will function identically (software will, of course, have to take account of the longer pipeline present and alter the stage programming accordingly).

1.9 MVNPP Latency

The latency of the MVNPP is a complex function that includes components from both the stage chip pipeline and the board logic external to the stage chips. The Cyto stage chip exhibits a latency of one line plus 17 pixels when processing an image and only two pixels when processing commands or when deactivated. Thus an N-stage pipeline of active stages will exhibit a latency of $N \times$ (one line $+ 17$)pixels. The Cyto stage latency is discussed in detail in the Stage Programmer's Manual. Over the VMEbus, the board logic adds two pixels of latency and over the Cyto-HSS or Stage Backplane buses, the board adds one pixel of latency to this total. Example number 3 (Section 3.3) discusses the MVNPP latency further with respect to image transfers over the VMEbus.

On the Maxbus the situation is more complex because both horizontal and vertical latency must be totaled separately. The FIFO at the end of the pipeline causes the vertical latency to round up one line and the horizontal latency to be set to one pixel. In addition, narrow images will cause horizontal latency of the stage chips to wrap around to vertical latency. To calculate the MVNPP vertical latency:

1. Sum the horizontal and vertical latencies for each stage in the pipeline and add one to the horizontal latency.
2. If the horizontal latency is greater than the actual image line-length, increment the vertical latency and subtract the line-length from the horizontal latency.
3. Repeat step 2 until the horizontal latency is less than the line-length.
4. Add one to the vertical latency (for the FIFO).

The result is the vertical latency from the input to output connectors of the MVNPP. The horizontal latency is always one pixel for Maxbus image transfers. Note that this calculation is not needed for CLbus transfers because there is no output from the MVNPP in this case.

2.0 Status and Control Registers

The MVNPP contains 16 words of VMEbus addressable I/O registers, only 14 of which are currently used. Figure 2 details the bit-level definitions of these registers. All of the registers are word accessible and, in addition, the pipeline image data register is byte and long-word accessible. The additional data widths available on the pipeline image data register allow increased processing performance with 32-bit CPUs transferring image data over the VMEbus.

2.1 Status Register - Offset 0

The status register contains six bit-fields: three error bits, two status bits and the board ID field. The following sections describe each bit-field and its behavior.

2.1.1 Board ID - Bits 7:0, Read-Only

This eight-bit field identifies this module as an MVNPP and indicates its revision level. Bits 7 through 2 are used to identify the board, and bits one and zero indicate a revision level with software differences. The MVNPP's ID number is 62 in decimal, and the current revision is zero. Refer to Appendix A for all revision-dependent programming information.

2.1.2 Pipeline Error - Bit 15, Read-Only

The pipeline error bit is set when either of the other board errors are set (see Sections 2.1.3 and 2.1.4). It is provided as a convenient single bit test for errors having occurred during processing. This bit is cleared by a VME reset or the Clear Errors bit in the control register (see section 2.2.3).

2.1.3 FIFO Overrun Error - Bit 14, Read-Only

The FIFO overrun error bit is set if the line buffer FIFO is written to when it is already full. This would normally be a software error caused by not reading data out of the pipeline; it only occurs when using the VMEbus for image I/O. This bit is cleared by a VME reset or the Clear Errors bit in the control register.

2.1.4 FIFO Underrun Error - Bit 13, Read-Only

The FIFO underrun error bit is set if the line buffer FIFO is read from when it is empty. This would normally be a software error caused by not writing data to the pipeline; it only occurs when using the VMEbus for image I/O. This bit is cleared by a VME reset or the Clear Errors bit in the control register.

2.1.5 Pipeline Collision Error - Bit 12, Read-Only

The pipeline collision error bit is set if the image I/O select bits (see Section 2.2.4) are switched while a high-speed image transfer is in progress. This would normally be a software

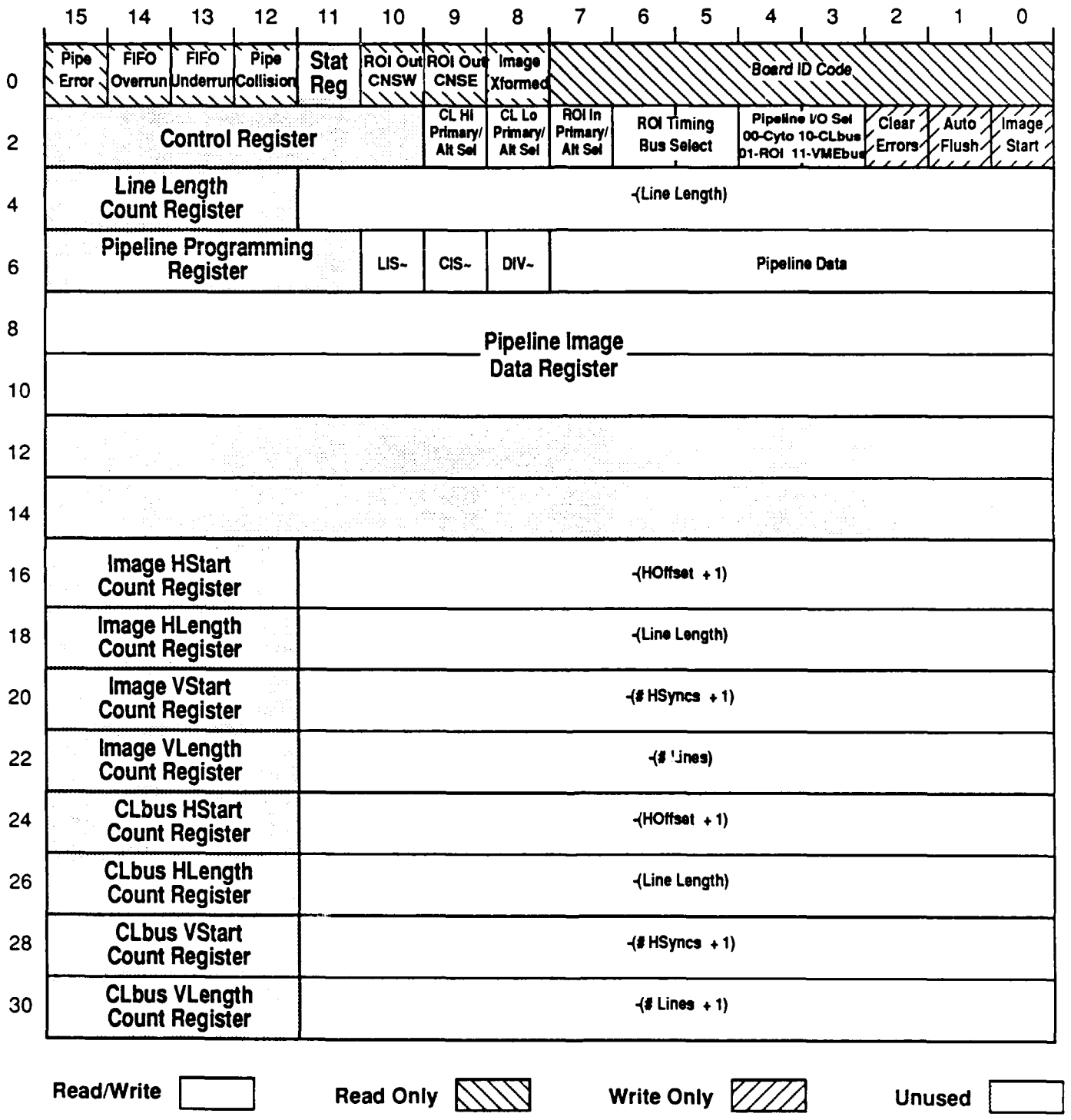

Figure 2 – MVNPP Control and Status Registers

error and would result in the garbling of high-speed data that was being processed by the MVNPP. This bit is cleared by a VME reset or the Clear Errors bit in the control register.

2.1.6 ROI Output Configuration Sense Bits - Bits 10:9, Read-Only

The ROI Output Configuration Sense Enable and Wire (CNSE and CNSW) bits are included to support Datacube's defined protocol for determining the data cable connections between a set of Maxbus processing boards. The CNSE status bit allows direct reading of the ROI output driver enable signal (which is a function of jumper on the board - see Section 4.1). This bit reads as a one if the driver is enabled and a zero if disabled. The CNSW bit allows direct reading of the primary/alternate select line driven from the input port connected to the MVNPP ROI output. The CNSW bit reads as a one if the output has been selected as a primary driver or a zero if it has been selected as an alternate driver. The Maxbus Specification (Datacube document number SP00-3), Section 1.2.3.3.2, describes an algorithm to determine the cable configuration of a Max Video system using these status bits and the ROI input Primary/Alternate Driver select bit (see Section 2.2.6).

2.1.7 Image Transformed - Bit 8, Read-Only

The image transformed bit provides the programmer a means to determine if the image was transformed by the previous circulation. This bit latches the stage chip XFRMN signal. See ERIM document IMT-86-461, "Integrated Neighborhood Processing Stage Design Specifications," for details of the operation of this signal. Briefly, this status bit will identify whether any of the stages in the pipeline transformed the image during the most recent image circulation. The image transformed bit is cleared at the start of an image transfer and by a VME system reset.

2.2 Control Register - Offset 2

This register contains six control bit fields that affect the operation of the MVNPP. The following sections describe these bits and their behavior in detail.

2.2.1 Image Start - Bit 0, write-only

Setting the image start bit performs several start-up functions necessary at the beginning of an image transfer via the VMEbus. This bit should be set just prior to the start of an image transfer over the VMEbus to eliminate extraneous data left over from previous transfers and to initialize the line-length counter. Specifically, setting this bit empties the line buffer FIFO, clears the image transformed status signal, and initializes the LIS~ generation circuitry.

2.2.2 Automatic Pipeline Flush - Bit 1, write-only

This bit, when set, enables automatic flushing of the stage pipeline during the final part of a VMEbus image transfer. The controlling software need only read the remainder of the image from the pipeline, and null data with the proper LIS~timing maintained will be automatically inserted at the pipeline input. This bit must be set after the last line of the

image is written to the pipeline. For proper operation, it must be cleared after the image read-out is complete. This bit is also cleared by a VMEbus reset.

2.2.3 Clear Errors - Bit 2, write-only

The clear errors bit, when set to a one, resets all the error bits in the MVNPP status register (see Section 2.1).

2.2.4 Pipeline I/O Select - Bits 4:3, Read-Write

These two bits select data from four different interfaces to the stage pipeline. Both the data and stage control signal drives are switched and any necessary control signal generation logic is also enabled. These bits are encoded to select from: Cyto-HSS (00), Datacube ROI (01), CLbus (10), and VMEbus (11) data transfers. The Cyto stage backplane interface is enabled by logic that detects when the MVNPP is inserted into a stage backplane. These bits are cleared (set to 00) by a VMEbus reset, which selects the Cyto-HSS interface by default.

2.2.5 ROI Timing Bus Select - Bits 6:5, Read-Write

These bits select which ROI timing bus (see the Datacube Maxbus Specification) will be used as a reference for data transferred through the stage pipeline. These bits are operative for both Datacube ROI and CLbus (see Section 2.2.4) data transfers. The two bits together form a binary number (00–11) that selects one of the HSYNC/VSYNC0–3~ . These bits are cleared (set to zero) by a VMEbus reset.

2.2.6 Datacube Input Primary/Alternate Driver Select - Bits 10:8, Read-Write

These bits select, for each Datacube input bus, either the primary or alternate output device connected to the input. The Maxbus allows two different processor outputs to be connected to a single input connector. The Primary/Alternate Driver Select bit for that input chooses one and only one of these outputs at a given time. This is intended to simplify the I/O cabling in a Maxbus system by allowing different logical processor connections to be selected under software control. Bits 8, 9, and 10 select the primary or alternate drivers for the ROI input bus, CLbus low byte, and CLbus high byte respectively. These bits are cleared (set to zero) by a VMEbus reset, which selects the primary driver by default.

2.3 Line Length Count Register - Offset 4, Bits 11:0, Read-Write

This register is used for the automatic generation of LIS~ during VMEbus image transfers through the pipeline. It must be loaded with the twos-complement of the number of pixels per line of the image to be transferred. The image start bit (see Section 2.2.1) must also be set immediately prior to the image transfer to initialize the line-length counter.

2.4 Pipeline Programming Data Register - Offset 6, Bits 10:0, Read-Write

This register provides access to the programming control bits of the stage pipeline. The VMEbus must be selected using the pipeline I/O select bits (see Section 2.2.4) before reading

or writing to this register. The programmer has complete control over DIV~ , CIS~ , and LIS~ in addition to data access. Only one byte at a time may be written or read to or from the pipeline using this register. During writes, DIV~ , CIS~ , and LIS~ must be supplied in bit positions 8 through 10. On reads, the current level of these control signals at the end of the pipeline is available on the same bits. The data read back comes directly from the end of the pipeline, not the line buffer FIFO as is the case with the pipeline image data register (see Section 2.5). Note that the output of the pipeline is returned by a read but the pipeline is only clocked by a write. Thus, two consecutive reads of this register will return identical data.

2.5 Pipeline Image Data Register - Offset 8,10, Bits 15:0,15:0, Read-Write

This register is the VMEbus image transfer access point to the Cyto stage pipeline. The VMEbus must be selected using the pipeline I/O select bits (see Section 2.2.4) before reading or writing to this register. Write accesses put data into the pipeline; read accesses get data from the line buffer FIFO at the end of the pipeline. The register is 32-bits wide and can be addressed as a long word, word, even byte or odd byte. Word access can be addressed to either word offset (8 or 10) interchangeably. Byte accesses may be addressed to any of the four bytes (offsets 8 through 11) of the register and will be properly routed to or from the pipeline input or output. LIS~ will be generated automatically for the image data written to this register, provided that the line length counter was correctly programmed and the image start bit (see section 2.2.1) was set prior to transferring image data.

2.6 ROI Image Transfer Timing Control Registers

Several timing counters are required to accurately identify and process only the valid image data from within a ROI image transfer. A ROI image transfer consists of a rectangular region of pixels surrounded (possibly on all four sides) by invalid pixels. The surrounding area of an ROI is defined by the VSYNC~ and HSYNC~ timing signals, but the ROI itself may occupy only a small region inside this larger area. Since the Cyto stage chip requires that only valid pixels be passed to it for processing, four counters must be defined to extract only the valid ROI from within the larger image area transferred. The following sections describe the required counters.

2.6.1 ROI Image Transfer HStart Register - Offset 16, Bits 11:0, Read-Write

These bits define the offset of the left edge of a ROI image relative to the HSYNC~ signal. This value is loaded into a counter at each HSYNC~ pulse and counted up to delay processing until valid data has arrived. The value written into this register must be the twos-complement of the number of pixels between the HSYNC~ pulse and the first valid pixel plus one (i.e., $-(\text{HOffset} + 1)$). The range of valid values is from (-1) to (-4096), which corresponds to horizontal offsets of from 0 to 4095 intervening pixels.

2.6.2 ROI Image Transfer HLength Register - Offset 18, Bits 11:0, Read-Write

These bits define the length of a line within the ROI image. The value loaded must be the twos-complement of the number of pixels contained in each line of the ROI image (i.e., −(Line Length)). This value is loaded into a counter when the HStart count value expires, and it counts off each valid pixel as it arrives. The range of valid values is (−1) to (−4096), which corresponds to line lengths of from 1 to 4096 pixels. Note, however, that the Cyto stage chip can currently only process images up to 2048 pixels wide.

2.6.3 ROI Image Transfer VStart Register - Offset 20, Bits 11:0, Read-Write

These bits define the top edge offset (in lines) of a ROI image relative to the VSYNC~ signal. This value is loaded into a counter at each VSYNC~ pulse and is counted up to delay processing until valid data has arrived. The value written into this register must be the twos-complement of the number of HSYNC~ s that will occur in between the VSYNC~ pulse and the first valid pixel, plus one (i.e., −(number of HSYNCs+1)). The range of valid values is from (−1) to (−4096) which corresponds to vertical offsets of from 0 to 4095 intervening lines.

2.6.4 ROI Image Transfer VLength Register - Offset 22, Bits 11:0, Read-Write

These bits define the number of lines within the ROI image. The value loaded must be the twos-complement of the number of lines contained in the ROI image (i.e., −(# Lines)). This value is loaded into a counter when the VStart count value expires and counts off each line (HSYNC~) as it arrives. The range of valid values is (-1) to (-4096) which corresponds to from 1 to 4096 lines in the ROI image.

2.7 CLbus Timing Control Registers - Offsets 24–30

Since the CLbus is based on ROIbus timing, the same requirements exist to extract valid data from the larger ROI transferred. A set of four counters analogous to the ROI Image Transfer Timing Control Registers of Section 2.6 are also needed and are illustrated in Figure 2. These registers operate identically to the counters of Section 2.6, including their value definitions and programming. These counters are active when the CLbus is selected as the high-speed I/O source for the MVNPP.

3.0 Programming Examples

In this chapter, examples of normal programming sequences for the MVNPP are described. The examples start with simple stage programming and progress through image circulation and subimage window circulation. Throughout the examples, hexadecimal numbers will be indicated with a 0x prefix. Also, the MVNPP board will be assumed to be addressable at 0x801300 in the VMEbus A32 address space.

The general programming sequence for using the MVNPP consists of: programming the stages, programming the MVNPP board registers, and recirculating an image through the pipeline. Programming the stages may be via the VMEbus pipeline programming port (see section 2.4) or one of the high-speed data ports. When using the VMEbus to program the stages, some large data movements can be performed more quickly using the VMEbus image data port. Programming the MVNPP board registers prepares the circuits for an image circulation.

3.1 Example 1: Sizing the Pipeline

This example will demonstrate how to determine how many stages are in the pipeline. This is an important parameter that is needed for programming the stages and correctly recirculating an image through the pipeline. The pipeline length may vary because multiple MVNPP boards can be cascaded transparently to the software. In addition, it is possible that failed stages could simply be bypassed in hardware on the MVNPP rather than replaced. This would result in a shorter than normal pipeline.

Pipeline sizing relies on the fact that the stages have a fixed latency of two clock cycles for propagating commands. Therefore, a command can be written to the pipeline and, by counting how many clocks are required for the command to appear at the pipeline output, the number of stages can be determined. To do this, the command and multiple null pixels are written to the pipeline programming port at address 0x801306. Each write, including the initial command, counts as one clock. Note that before any writes to the pipeline can be performed, the pipeline I/O select bits (see Section 2.2.4) must be set to select VMEbus access of the pipeline via a word write of 0x0018 to address 0x801302. This value should also be logically (bit-wise) ORed with any value that is later written to the control register so long as VMEbus access to the pipeline is desired.

The command written should be the word 0x0500, which corresponds to a Global Activate stage command with the CIS~ control signal active and the LIS~ and DIV~ control signals inactive. The null pixels are 0x07XX, where the XX is a "don't care" bit pattern. After each word is written, the programming data port should be read. If the value read matches the initial command word written (with unused bits masked off), then the command has propagated through the pipeline. The total number of words written to the programming port, divided by two, yields the number of stages in the pipeline. Note that there is an extra

register at the end of the stage pipeline on the MVNPP board. This extra register's latency will force the number of clocks, before the command comes out of the pipeline, to be one higher than twice the number of stages.

A maximum length error check should also be included in the stage sizing loop. This will prevent any hardware problem from causing the program to loop forever. The value should be set high (perhaps 1000) to ensure that a long pipeline will not be flagged as broken.

3.2 Example 2: Programming the Stages Over the VMEbus

Programming the Cytocomputer stage chips is a complex subject that is covered in detail in the document "Stage Programmer's Manual," ERIM document number IPTL-89-294. This example abstracts that process to the following three steps: (1) send stage command preamble, (2) send stage program, (3) send stage command post-amble. The preamble and post-amble are written to the MVNPP pipeline programming port to allow setting of the stage control bits. The stage program may be sent through the image data port for a higher transfer rate.

Before any stage commands are written, select VMEbus pipeline I/O by writing 0x0018 to the control register. The stage command preamble must be written to the pipeline programming port (at address 0x801306) with the DIV~ bit active (i.e., cleared) for command bytes and inactive for the null bytes between commands. The actual stage programs can be lengthy (several hundred bytes), and a significant speedup in programming can be realized by using the image data port (at address 0x801308) to write the program data to the stages. The speedup is due to the longword accesses available through the image data port. Stage program writes through the programming port proceed at only one byte per access. After the stage program is written, the stage post-amble must be written to the programming port with the CIS~ bit active for command bytes.

The above sequence will be repeated for each stage to be used during a circulation. Many times not all stages available in the pipeline will be needed for a given circulation. The remaining stages must be commanded to ignore the image (i.e., deactivated) as it passes through them. These commands (one for each remaining stage) are written to the programming port after all the stages to be used have been programmed. The deactivate command will trickle through the pipeline to the last stages in front of the image data.

3.3 Example 3: Image Circulation Over the VMEbus

This example continues Example 2 by showing how to circulate an image through the programmed stages. After the stages have been programmed (via Example 2), the image line-length counter must be set. The twos-complement of the actual image line length must be written to address 0x801304. Next, the image start bit must be set with a word write of 0x0019 to address 0x801302 (0x0001 ORed with 0x0018). This initializes the MVNPP for a VMEbus image circulation and starts the automatic LIS~ generation.

The actual image circulation is made up of three steps for most images (or three slightly different steps for very small images). The normal sequence is:

1. Write initial lines of the image to fill the stage pipeline.
2. Alternately write and read image lines to and from the pipeline until the whole image is written,
3. Set the image flush bit and read the remaining image lines from the pipeline.

In Step 1 above, the stage internal pixel latency (one image line plus 17 pixels) is filled until between one pixel and one line has entered the FIFO at the end of the pipeline. This must be done so that there will be image data available in the FIFO before it is first read. Step 2 is the main data movement loop of the image circulation. Alternately writing and reading image lines provides good data movement efficiency and prevents the FIFO from overflowing. Note that the FIFO is large enough to hold at least two image lines up to the maximum line length support by the Cyto-HSS stage. In Step 3, the image pixels remaining in the pipeline after the last line is written are read out.

The sequence above applies if the image has enough lines to fill the pipeline and put at least one pixel in the FIFO. If this is not the case, then the following sequence must be used:

1. Write the entire image to the pipeline.
2. Set the image flush bit and and continue writing pixels until at least one pixel is in the FIFO.
3. Read the image from the FIFO.

This sequence ensures that data will have reached the FIFO before it is read.

The complete latency of the pipeline can be determined from the pipeline length and the number of stages programmed to take part (activated) in a circulation. Example 1 illustrated how to determine the pipeline length. It is assumed here that the programmer can determine how many stages he/she programmed and deactivated. The formula below gives the pipeline latency:

$$\text{Latency} = (\text{Number of Active Stages}) \times (\text{Line Length} + 17) + 2(\text{Number of Inactive Stages}) + 2$$

Where the Number of Inactive Stages is the pipeline length minus the Number of Active Stages (this formula is accurate only for VMEbus data circulations through the pipeline). The additional two pixels added to the end account for MVNPP hardware latency in addition to the stage chips. Note that latency for image transfers is longer than the latency when accessing the VMEbus programming port because the FIFO buffer adds one clock cycle of latency.

3.4 Minimizing VMEbus Image Circulation Time

This section will focus on the performance issue of minimizing the time required to circulate an image over the VMEbus. The VMEbus performance limit can easily be surpassed by using one of the high-speed I/O buses supported by the MVNPP, but this may not be a cost-effective solution. A nonoptimal coding for the VMEbus image circulation routine could slow the image circulation from 2 times to 20 times or more. Two main areas must be addressed to improve the image circulation times: (1) low-level data movement, and (2) image storage boundaries.

For larger sized images, most of the circulation time will be spent alternately writing and reading lines of the image to and from the pipeline image data port. To minimize this time a tight loop, perhaps hand coded in assembly language, should be used. This loop should use long-word moves even if the CPU doesn't have a 32-bit data bus (e.g., the 68010), because this will minimize instruction fetching per byte moved. In addition, if the image line length is less than 2048, then two or more lines may be written or read at once to minimize the overhead of changing from writing to reading and back. The only limit is that the FIFO must not be overrun or underrun while writing or reading data.

To make long-word accesses effective on a machine with 32-bit data paths, each image line should be stored on a long-word boundary and be a multiple of four pixels long. Many times, however, this is not possible due to arbitrary line lengths or windowed processing within an image. In these cases, up to three bytes at the beginning and/or end of the image line must be moved before a long-word boundary occurs. A simple method to optimally accomodate these cases would be to code four different data movement routines and, on a line-by-line basis, call the appropriate movement routine for each case for writing or reading. The movement routine would move the preamble bytes (if necessary), move the majority of the line with long words, and then move the trailing bytes (if necessary).

3.5 Example 4: Cyto-HSS Image Circulation

This example will demonstrate how to use the Cyto-HSS high-speed I/O interface. This interface may be used to transfer an image through the pipeline or to program the stages. The Cyto-HSS I/O must be selected via a write of 0x0000 to the Control Register at address 0x801302. This will select the Cyto-HSS connectors as the source and destination of the stage pipeline. This condition is the power-on default for the MVNPP and therefore, may not need any selection programming. Now the MVNPP will appear to the Cyto-HSS system exactly as a pipeline of Cyto chip stages for all normal processing functions (see Section 1.4), and may be programmed or used in processing identically to the existing hardware. The programmer must ensure that any writes to the control register continue to select the Cyto-HSS for pipeline I/O if further processing will be performed.

3.6 Example 5: Maxbus Image Circulation

This example will show how to use the Maxbus high-speed I/O interface to transfer data through the stage pipeline. Assuming that the stages have already been programmed and are

ready to process an image, the amount of initialization required is minimal. First, the four ROI image count registers must be initialized. These counters define to the MVNPP exactly where there are valid pixels inside the rectangular region defined by the Maxbus VSYNC~ and HSYNC~ . This is important because the Cyto stage utilizes off-image information in its processing functions. The image HLength and VLength registers must contain the twos-complement of the number of pixels per line and the number of lines in the image. These must be the number of lines in the actual image of interest and not necessarily the values programmed into a ROIstore. The HStart and VStart registers must contain the number of pixels or lines that appear between the HSYNC~ or VSYNC~ and the first valid pixel.

As an example, assume the ROI to be transferred is 300 pixels wide and 500 lines long and that the valid image data within this is 256 pixels wide and 400 pixels long and will start on the 8th line and 30th pixel. In this case the following four values would be written to the ROI Image Timing Control Registers:

Register	Value
HStart Register	0xffe3
HLength Register	0xff00
VStart Register	0xfff8

Note that the HStart and VStart register values are the twos-complement of the pixel and line offset plus one. Also note that these offset values must be determined based on the horizontal and vertical latency of the Maxbus boards between the MVNPP and the ROIstore supplying the image. The Datacube manuals supplied with each board should describe how to calculate the board's latency.

Once the ROI counter registers are programmed, the ROI image I/O connectors, the ROI timing bus, and the primary or alternate source for the input must all be selected. This is accomplished by writing to the MVNPP Control register at address 0x801302. If, for instance, ROI timing bus 2 was to be used along with the alternate source connected to the ROI image input connector, then a value of 0x00c8 would be written to the Control register. This value would select the above two parameters and continue to enable the ROI image I/O to access the stage pipeline. At this point the MVNPP is ready to process an image and the ROIstore supplying the data could be initialized to transfer an image in either single-shot or continuous mode.

3.7 Example 6: CLbus Programming of the Pipeline

This example will show how to use the CLbus interface to quickly program a pipeline of Cyto-HSS stage chips. The CLbus interface uses two Maxbus data buses to supply a data byte and the three stage-control bits to the pipeline on the MVNPP. The two Maxbus paths are called the CLbus Lo and CLbus Hi data paths. The CLbus Lo data path connects to the pipeline data input. The low-order three bits of the CLbus Hi data pash connect to the pipeline control bits in the same order as in the Pipeline Programming Register (see Section 2.3). Therefore, the same sequence of data words written to the Pipeline Programming

Register can be placed in a ROIstore memory for transmission over the CLbus to program the pipeline. These words must be placed in the ROIstore so that the high-order byte of the word (the byte with the stage control bits) appears on the CLbus Hi data path during the CLbus data transfer. Note that the stage programming data does not have any inherent image line structure; the ROI line length for the transfer may be chosen arbitrarily as convenient. However, the data must be padded with null pixels to fill out the last valid line of the ROI data transfer. The Stage Programmer's Manual describes in detail the actual data and control signal values that must be written to correctly program the stage pipeline.

In addition to defining the pipeline programming data, the MVNPP CLbus Timing Control registers must be programmed similarly to Example 5. The data transfer that programs the pipeline appears to be a ROI image transfer to the rest of the system. Because of this, the CLbus Timing Control registers are used to define exactly which data within the transfer are valid programming data for the pipeline. The chosen ROI image line length and number of lines are written to the CLbus HLength and VLength registers (in twos-complement form). The CLbus HStart and VStart registers are likewise programmed according to the latency of the system boards prior to the MVNPP in the pipeline.

Finally, the MVNPP Control register must be programmed to select the CLbus input, ROI timing bus, and primary or alternate drivers for the CLbus inputs. If the primary drivers and ROI timing bus number 1 are desired, then a value of 0x0030 would be written to address 0x801302. At this point, the MVNPP is ready to receive pipeline programming data over the CLbus. The ROIstore(s) that will supply the data may be initialized to start transferring the data.

4.0 MVNPP Configuration

Configuring the MVNPP consists of setting jumper blocks to the desired configuration, installing the board in a VMEbus chassis, and connecting appropriate cabling to other cards in the system. There are three jumper blocks and one sixteen-position rotary switch on the revision zero MVNPP board. Figure 3 illustrates these configuration blocks' locations and their standard configurations. The figure also shows the location of each of the connectors on the MVNPP and their names and sizes. The following sections describe each of these items and their usage.

4.1 Internal Board Configuration

Rotary switch SW1, in conjunction with two address decoding PALs, determine the MVNPP control registers' base address in the VMEbus address space. The PALs are normally programmed to set the base address at 0x801300 for A24 and A32 accesses and at 0x1300 for A16 accesses. Switch SW1 allows the user to select this address or 0x1320, 0x1340, or 0x1360 via its 0, 2, 4, or 6 setting. Since only four address selections are available, the other positions of SW1 wrap around in binary fashion to also select one of the addresses. By default, SW1 is set to 0 by ERIM to select address 0x1300.

Jumper J1 configures the Maxbus output data path as a primary, alternate, or always-enabled output. This jumper may be configured to match the user's need in a Maxbus-based system. Positions A, B, and C correspond to the primary, alternate, and always-enabled states respectively. By default, the jumper is installed in the A position by ERIM to select a primary output path.

Jumpers J2 and J3 allow bypassing of a delay in the clock path for the Cyto-HSS and stage backplane I/O interfaces. They have no effect when any other pipeline I/O interfaces are used. This delay should only be be bypassed when the MVNPP board is used in connection with an MVICP combiner board. Position A of each jumper bypasses the delay. Position B of each jumper inserts the delay. By default, these jumpers are set by ERIM to the correct position for the particular system in which the MVNPP board will reside.

In addition to the jumpers and switches, there are two resistor packs (S2 and S3) that may need to be installed based on the user's system connection. These are termination resistors to the Maxbus ECL timing bus control signals. They should be installed only if the MVNPP is at the physical end of the ECL timing cable. The terminators consist of a 10-pin and an 8-pin package that are placed in two sockets adjacent to connector P3 (see Section 4.2) on the MVNPP board. The 10-pin package must be installed with pin number 1 (indicated by a colored dot on the package edge opposite the pins) in the socket pin labeled for it. The 8-pin package may be installed in either orientation. By default the terminators are installed by ERIM and must be removed if the MVNPP is not at the end of the Maxbus ECL cable.

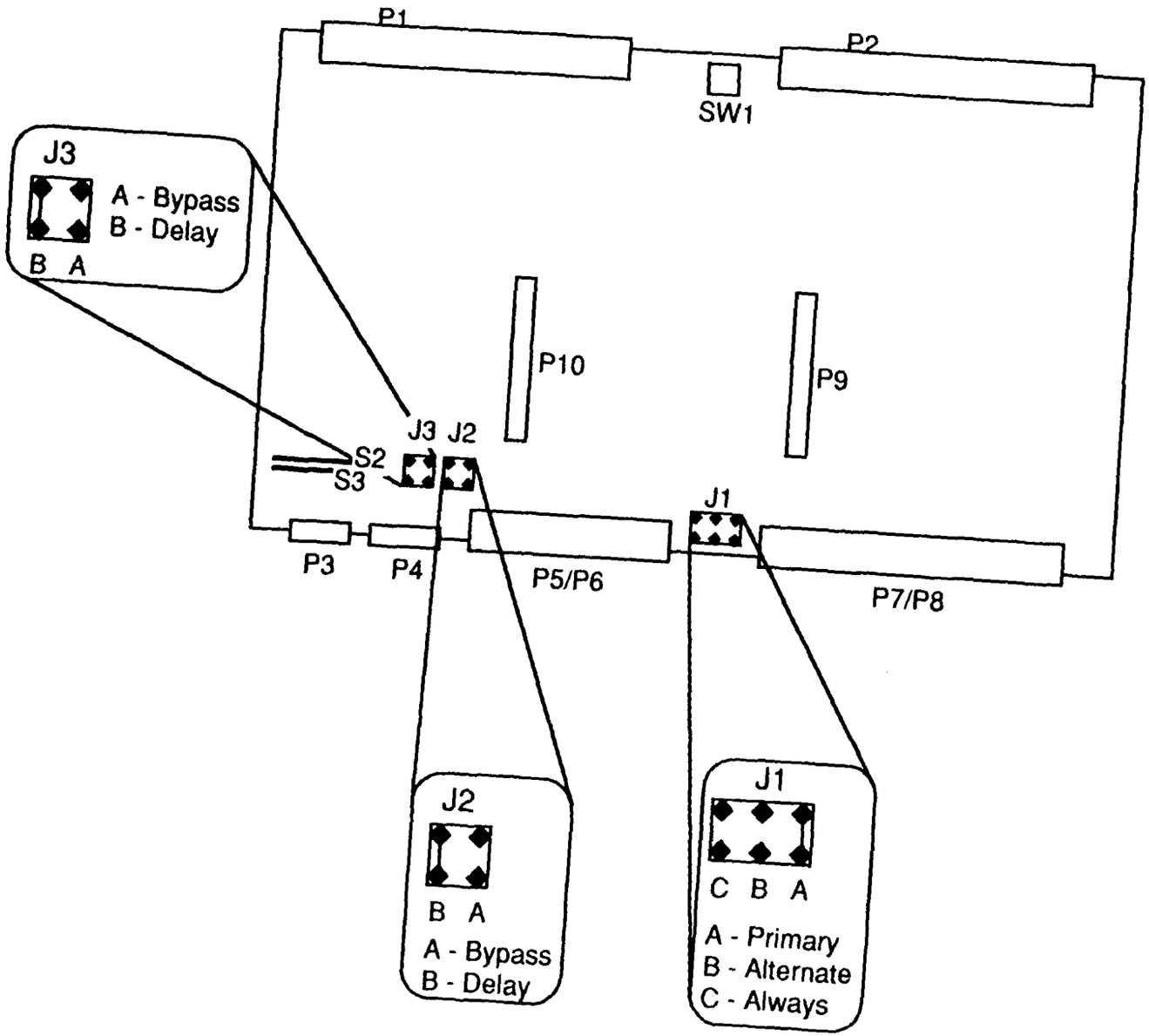

Figure 3. MVNPP Board Configuration Locations

4.2 Connector Functions

The P1 and P2 connectors are standard VMEbus connectors whose pinouts conform to the VMEbus Specification Revision C (IEEE P1014/D1.0). The outer two rows of the P2 connector contain many ERIM defined signals that may conflict with other defined buses such as the VSB extension bus. Pinouts for these signals and the additional connectors appear in Figures 4 through 7. The following paragraphs describe these connectors and their functions in the order the user will most likely have to use them.

Connectors P9 and P10 (see Figure 4) provide the physical connection to a daughter card that can hold up to 12 Cytocomputer chip stages. These connectors are asymmetrically placed to ensure that the daughter card will be correctly installed on the MVNPP main board. The daughter card should be installed so that its top and bottom edges align with the main-board's edges. The supplied plastic spacers should be installed through the holes at each corner of the daughter card to support it. If the daughter card will not be installed (or was not purchased), then jumpers must be installed across each pair of pins in connector P9. These jumpers connect signals that would otherwise be connected on the daughter card itself. These jumpers are installed by ERIM if the MVNPP board is purchased without a daughter card.

Connector P2 (see Figure 5) contains signals on its outer two rows that allow the MVNPP to be directly plugged into a Cyto-HSS stage backplane. This can provide up to 320 stages per card cage in a Cyto-HSS workstation. No jumpers need be set to enable operation; the MVNPP automatically detects the stage backplane and configures itself for that environment. Note that because of these signals' presence, the MVNPP must be plugged into a VMEbus slot that does not contain any other conflicting signal definitions on P2 rows A or C for correct operation.

Connectors P3 and P4 (see Figure 6) are the standard Maxbus ECL and ROI timing bus connectors. These timing bus cables may be bused through the MVNPP connectors in the same way as any other Maxbus-based board. Refer to the Maxbus Specification manual for additional details.

Connectors P5 and P6 (see Figure 6) form a dual 26-pin connector stack at the board edge and are labeled P5/P6. Connector P6 contains both a Maxbus input and output data path on the single 26-pin connector and is the bottom connector. A cable supplied with the MVNPP separates the input and output paths and provides a Maxbus standard 14-pin connector to plug into the user's other Maxbus boards. Each connector of the cable is labeled appropriately as P6, MVNPP In, or MVNPP Out. Note that the MVNPP In connector should be plugged into an <u>output</u> port of some other Maxbus board and the MVNPP Out connector should be plugged into an <u>input</u> port of some other Maxbus board. Connector P5 is identical to P6, but it contains the two CLbus input data paths. A second split cable is supplied with the MVNPP that is labeled for P5 and the CLbus Hi and Lo data paths. Both 14-pin connectors on this cable should be plugged into <u>output</u> ports of other Maxbus boards.

Connector P9

	Daughter Card Signal		Daughter Card Signal
1	VCC	2	VCC
3	VCC	4	VCC
5	VCC	6	VCC
7	P2_D0	8	P14_D0
9	P2_D1	10	P14_D1
11	P2_D2	12	P14_D2
13	P2_D3	14	P14_D3
15	P2_D4	16	P14_D4
17	P2_D5	18	P14_D5
19	P2_D6	20	P14_D6
21	P2_D7	22	P14_D7
23	GND	24	GND
25	P2_DOV~	26	P14_DOV~
27	P2_LOS~	28	P14_LOS~
29	P2_COS~	30	P14_COS~
31	GND	32	GND
33	STG_CLK_IN	34	STG_CLK_OUT
35	GND	36	GND
37	GND	38	GND
39	HIGH	40	DCARD_THERE~

Connector P10

	Daughter Card Signal		Daughter Card Signal
1	S3_PINACT	2	S3_PIMAGE
3	S4_PINACT	4	S4_PIMAGE
5	S5_PINACT	6	S5_PIMAGE
7	S6_PINACT	8	S6_PIMAGE
9	S7_PINACT	10	S7_PIMAGE
11	S8_PINACT	12	S8_PIMAGE
13	S9_PINACT	14	S9_PIMAGE
15	S10_PINACT	16	S10_PIMAGE
17	S11_PINACT	18	S11_PIMAGE
19	S12_PINACT	20	S12_PIMAGE
21	S13_PINACT	22	S13_PIMAGE
23	S14_PINACT	24	S14_PIMAGE
25	GND	26	GND
27	S3_XFORM~	28	S4_XFORM~
29	S5_XFORM~	30	S6_XFORM~
31	S7_XFORM~	32	S8_XFORM~
33	S9_XFORM~	34	S10_XFORM~
35	S11_XFORM~	36	S12_XFORM~
37	S13_XFORM~	38	S14_XFORM~
39	GND	40	GND

Figure 4. MVNPP Connectors P9 and P10 Pinouts

	Row A Signals	Row B Signals	Row C Signals
1	GND	+5V	GND
2	CCK	GND	GND
3	GND	RESERVED	GND
4	N.C.	A24	GND
5	GND	A25	GND
6	SPI~	A26	SPO~
7	LSO~	A27	LIS~
8	SR_DOV~	A28	SR_LOS~
9	SR_COS~	A29	RESERVERD
10	SR_D1	A30	SR_D0
11	SR_D3	A31	SR_D2
12	SR_D5	GND	SR_D4
13	SR_D7	+5V	SR_D6
14	+5V	D16	+5V
15	SI_DIV~	D17	SO_DOV~
16	SI_LIS~	D18	SO_LOS~
17	SI_CIS~	D19	SO_COS~
18	SI_D0	D20	SO_D0
19	SI_D1	D21	SO_D1
20	SI_D2	D22	SO_D2
21	SI_D3	D23	SO_D3
22	SI_D4	GND	SO_D4
23	SI_D5	D24	SO_D5
24	SI_D6	D25	SO_D6
25	SI_D7	D26	SO_D7
26	N.C.	D27	N.C.
27	N.C.	D28	N.C.
28	N.C.	D29	N.C.
29	+5V	D30	+5V
30	+5V	D31	+5V
31	GND	GND	GND
32	GND	+5V	GND

Figure 5. MVNPP Connectors P2 Pinout

Connector P3

	Maxbus ECL Signal		Maxbus ECL Signal
1	E2DC	2	E2DC~
3	GND	4	E4DC
5	E4DC~	6	GND
7	EHR	8	EHR~
9	EVR	10	EVR~

Connector P4

	ROI Timing Signal		ROI Timing Signal
1	RESERVED	2	ROI_VSYNC3~
3	ROI_HSYNC3~	4	GND
5	ROI_VSYNC2~	6	ROI_HSYNC2~
7	GND	8	ROI_VSYNC1~
9	ROI_HSYNC1~	10	GND
11	ROI_VSYNC0~	12	ROI_HSYNC0~
13	GND	14	RESERVED

Connector P5

	CLbus Input Signal		CLbus Input Signal
1	CLHI_ALT_PRIMARY~	2	CL_D15
3	CL_D14	4	GND
5	CL_D13	6	CL_D12
7	GND	8	CL_D11
9	CL_D10	10	GND
11	CL_D9	12	CL_D8
13	GND	14	CLLO_ALT_PRIMARY~
15	CL_D7	16	CL_D6
17	GND	18	CL_D5
19	CL_D4	20	GND
21	CL_D3	22	CL_D2
23	GND	24	CL_D1
25	CL_D0	26	GND

Connector P6

	Maxbus I/O Signal		Maxbus I/O Signal
1	MI_ALT_PRIMARY~	2	MI_D7
3	MI_D6	4	GND
5	MI_D5	6	MI_D4
7	GND	8	MI_D3
9	MI_D2	10	GND
11	MI_D1	12	MI_D0
13	GND	14	MO_ALT_PRIMARY~
15	MO_D7	16	MO_D6
17	GND	18	MO_D5
19	MO_D4	20	GND
21	MO_D3	22	MO_D2
23	GND	24	MO_D1
25	MO_D0	26	GND

Figure 6. MVNPP Maxbus Connector Pinouts (P3 Through P6)

Connectors P7 and P8 (see Figure 7) form a dual 40-pin connector stack at the board edge and are labeled P7/P8. These two connectors allow the MVNPP to be directly connected to a Cyto-HSS port and combiner in a Cyto-HSS workstation. They also allow multiple MVNPP boards to be physically cascaded to create a longer pipeline. Different cables are supplied with the MVNPP to accomplish these two functions. Connector P7 is the bottom one in the stack and contains the Cyto-HSS input path signals. A cable is supplied and appropriately labeled to link P7 with the Cyto-HSS port. Connector P8 contains the Cyto-HSS output path signals, and a cable is also supplied to link P8 with the Cyto-HSS combiner.

A third cable is also supplied to cascade two MVNPP boards. To cascade boards in a Cyto-HSS system, the port would link to P7 of the first MVNPP; the cascade cable would link P8 of the first MVNPP board to P7 of the second MVNPP board; and P8 of the second MVNPP board would be linked to the combiner. If more than two MVNPPs are present, the additional boards would be daisy-chained, P8 to P7, similar to the two-board method. For a Maxbus system, the cascade cable placement would be identical, but the supplied Maxbus cables would be plugged into connectors P5 and P6 on the first board (for input ports) and the last board (for output ports). Note that this will require a cable plugged into connector P5 on the first and last boards, with only one of the 14-pin ends connected on each cable. An extra P5 cable is supplied for multiboard MVNPP purchases.

Connector P7

	HSS Input Signal		HSS Input Signal
1	GND	2	GND
3	GND	4	GND
5	GND	6	HI_D0
7	GND	8	HI_D1
9	GND	10	HI_D2
11	GND	12	HI_D3
13	GND	14	HI_D4
15	GND	16	HI_D5
17	GND	18	HI_D6
19	GND	20	HI_D7
21	GND	22	GND
23	GND	24	HSS_CLK
25	GND	26	GND
27	GND	28	HI_DIV~
29	GND	30	HI_LIS~
31	GND	32	HI_CIS~
33	GND	34	GND
35	GND	36	STG_CLK_EN~
37	LSO~	38	SPI~
39	GND	40	XFORM_IN~

Connector P8

	HSS Output Signal		HSS Output Signal
1	GND	2	GND
3	GND	4	GND
5	GND	6	HO_D0
7	GND	8	HO_D1
9	GND	10	HO_D2
11	GND	12	HO_D3
13	GND	14	HO_D4
15	GND	16	HO_D5
17	GND	18	HO_D6
19	GND	20	HO_D7
21	GND	22	GND
23	GND	24	HSS_CLK
25	GND	26	GND
27	GND	28	HO_DOV~
29	GND	30	HO_LOS~
31	GND	32	HO_COS~
33	GND	34	GND
35	GND	36	STG_CLK_EN~
37	LSI~	38	SPO~
39	GND	40	STG_XFORM~

Figure 7. MVNPP Connectors P7 and P8 Pinouts

Appendix: Revision Dependent Information

Both wire-wrap prototype and production versions of the MVNPP have been built. Chapters I through IV fully describe the production version of the board; difference in the wire-wrap version are described below. There are no programming differences between the two versions, but there are physical and cabling differences. First, the wire-wrap board cannot accept a daughter card and so is limited to four stage chips in its pipeline. In addition, the jumper and connector positions on the wire-wrap board are different than on the production board. Figure A–1 identifies the jumper and connector positions on the wire-wrap board. Note that there are daughter card connector pins defined even though no daughter card can be inserted. This was done to facilitate board debugging.

The front panel I/O connectors are the major difference between the wire-wrap and production board. Two 40-pin connectors (P5 and P6) take the place of connectors P5 through P8 on the production board version. The Maxbus and Cyto-HSS I/O signals are multiplexed on these two connectors as shown in Figure A–2. This arrangement was required by the limited connector area of the wire-wrap board itself. This arrangement prevents two wire-wrap MVNPP boards from being cascaded if they are used in a Maxbus-based system. The wire-wrap board also requires different cable assemblies than the production board for connection to other Maxbus boards.

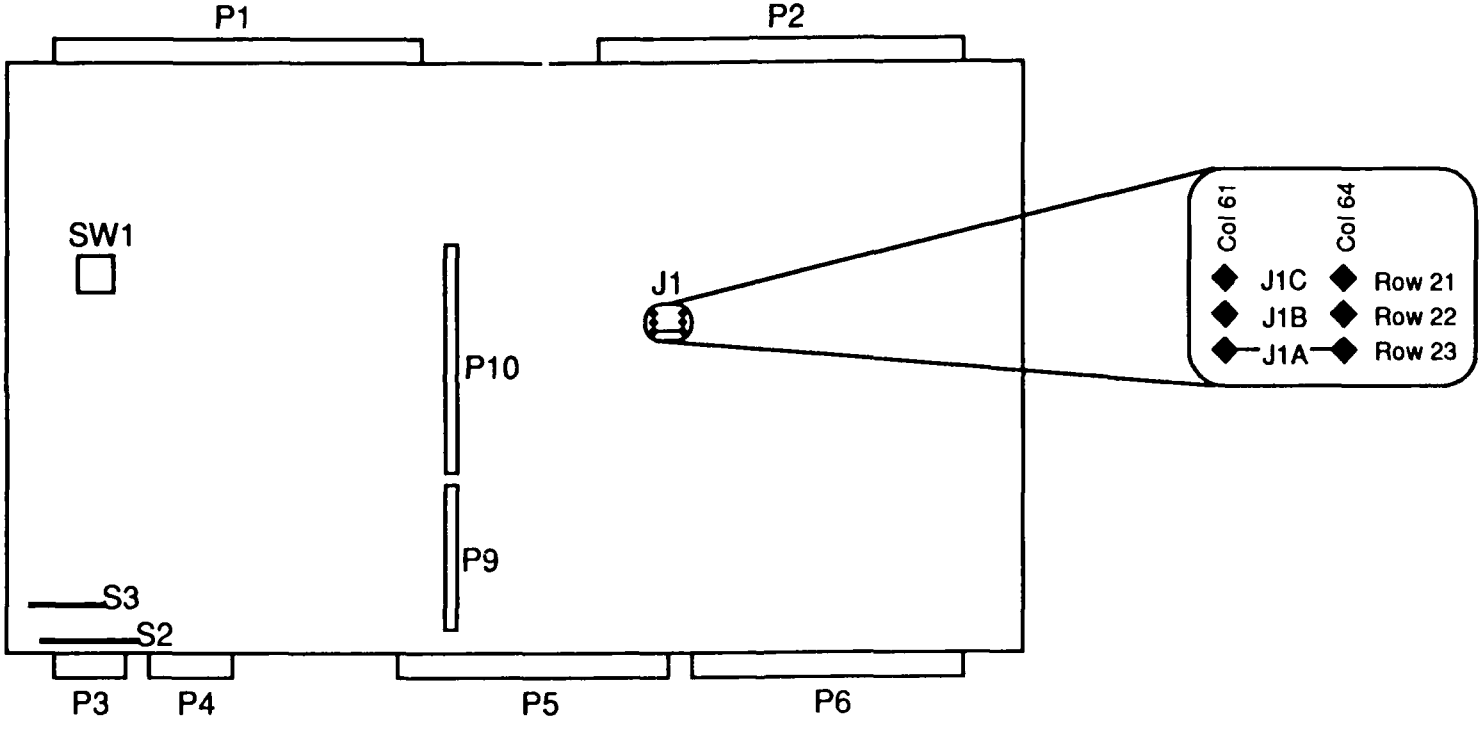

Figure A–1. MVNPP Wire-Wrap Board Configuration Locations

Connector P5

	HSS Input Signal	CLbus Input Signal		HSS Input Signal	CLbus Input Signal
1		CLHI_ALT_PRIMARY~	2		CL_D15
3		CL_D14	4		
5		CL_D13	6	HI_D0	CL_D12
7			8	HI_D1	CL_D11
9		CL_D10	10	HI_D2	
11		CL_D9	12	HI_D3	CL_D8
13			14	HI_D4	
15		CLLO_ALT_PRIMARY~	16	HI_D5	CL_D7
17		CL_D6	18	HI_D6	
19		CL_D5	20	HI_D7	CL_D4
21			22		CL_D3
23		CL_D2	24	HSS_CLK	
25		CL_D1	26		CL_D0
27			28	HI_DIV~	
29			30	HI_LIS~	
31			32	HI_CIS~	
33			34		
35			36	STG_CLK_EN~	
37	LSO~		38	SPI~	
39			40	XFORM_IN~	

Connector P6

	HSS Output Signal	Maxbus I/O Signal		HSS Output Signal	Maxbus I/O Signal
1		MI_ALT_PRIMARY~	2		MI_D7
3		MI_D6	4		
5		MI_D5	6	HO_D0	MI_D4
7			8	HO_D1	MI_D3
9		MI_D2	10	HO_D2	
11		MI_D1	12	HO_D3	MI_D0
13			14	HO_D4	
15		CLLO_ALT_PRIMARY~	16	HO_D5	MO_D7
17		MO_D6	18	HO_D6	
19		MO_D5	20	HO_D7	MO_D4
21			22		MO_D3
23		MO_D2	24		
25		MO_D1	26		MO_D0
27			28	HO_DOV~	
29			30	HO_LOS~	
31			32	HO_COS~	
33			34		
35			36	STG_CLK_EN~	
37	LSI~		38	SPO~	
39			40	STG_XFORM~	

Figure A–2. MVNPP Wire-Wrap Connectors P5 and P6 Pinout